

# Improved Residual Vector Quantization for High-dimensional Approximate Nearest Neighbor Search

Liu Shicong, Lu Hongtao, Shao Junru

{artheru, yz\_sjr, htlu}@sjtu.edu.cn Shanghai Jiaotong University

## Abstract

Quantization methods have been introduced to perform large scale approximate nearest search tasks. Residual Vector Quantization (RVQ) is one of the effective quantization methods. RVQ uses a multi-stage codebook learning scheme to lower the quantization error stage by stage. However, there are two major limitations for RVQ when applied to on high-dimensional approximate nearest neighbor search: 1. The performance gain diminishes quickly with added stages. 2. Encoding a vector with RVQ is actually NP-hard. In this paper, we propose an improved residual vector quantization (IRVQ) method, our IRVQ learns codebook with a hybrid method of subspace clustering and warm-started k-means on each stage to prevent performance gain from dropping, and uses a multi-path encoding scheme to encode a vector with lower distortion. Experimental results on the benchmark datasets show that our method gives substantially improves RVQ and delivers better performance compared to the state-of-the-art.

## Introduction

Nearest neighbor search is a fundamental problem in many computer vision applications such as image retrieval (Rui, Huang, and Chang 1999) and image recognition (Lowe 1999). In high dimensional data-space, nearest neighbor search becomes very expensive due to the curse of dimensionality (Indyk and Motwani 1998). Approximate nearest neighbor (ANN) search is a much more practical approach. Quantization-based algorithms have recently been developed to perform ANN search tasks. They achieved superior performances against other ANN search methods (Jegou, Douze, and Schmid 2011). Product Quantization (Jegou, Douze, and Schmid 2011) is a representative quantization algorithm. PQ splits the original  $d$ -dimensional data vector into  $M$  disjoint sub-vectors and learn  $M$  codebooks  $\{C_1 \cdots C_M\}$ , where each codebook contains  $K$  codewords  $C_m = \{c_m(1), \cdots, c_m(K)\}, m \in 1 \cdots M$ . Then the original data vector is approximated by the Cartesian product of the codewords it has been assigned to. PQ allows fast distance computation between a quantized vector  $\mathbf{x}$  and an input query vector  $\mathbf{q}$  via asymmetric distance

computation (ADC): the distances between  $\mathbf{q}$  and all codewords  $c_m(k), m \in 1 \cdots M, k \in 1 \cdots K$  are precomputed, then the approximate distance between  $\mathbf{q}$  and  $\mathbf{x}$  can be efficiently computed by the sum of distances between  $\mathbf{q}$  and codewords of  $\mathbf{x}$  in  $O(M)$  time. Compared to the exact distance computation taking  $O(d)$  time, the time complexity is drastically reduced.

Product Quantization is based on the assumption that the sub-vectors are statistically mutual independent, such that the original vector can be effectively represented by the Cartesian product of quantized sub-vectors. However vectors in real data do not all meet that assumption. Optimized Product Quantization (OPQ) (Ge et al. 2013) and Cartesian K-means (Norouzi and Fleet 2013) are proposed to find an optimal subspace decomposition to overcome this issue.

Residual Vector Quantization (RVQ) (Chen, Guan, and Wang 2010) is an alternative approach to perform approximate nearest neighbor search task. Similar to Additive Quantization (AQ) (Babenko and Lempitsky 2014) and Composite Quantization (Ting Zhang 2014), RVQ approximates the original vector as the sum of codewords instead of Cartesian product. Asymmetric distance computation can also be applied to data quantized by RVQ. RVQ adopts a multi-stage clustering scheme, on each stage the residual vectors are clustered instead of a segment of the original vector. Compared to PQ, RVQ naturally produces mutually independent codebooks. However, the gain of adding an additional stage drops quickly as residual vectors become more random, limiting the effectiveness of multi-stage methods to only a few stages (Gersho and Gray 1992). A direct observation is that the encodings of codebooks learned on the latter stages have low information entropy. Moreover, encoding a vector with dictionaries learned by RVQ is essentially a high-order Markov random field problem, which is NP-hard.

In this paper, we propose the Improved Residual Vector Quantization (IRVQ). IRVQ uses a hybrid method of subspaces clustering and warm-started k-means to obtain high information entropy for each codebook, and uses a multi-path search method to obtain a better encoding. The basic idea behind IRVQ is rather simple:

1. Subspace clustering generally produces high information entropy codebook. Though we seek a clustering on the whole feature space, such codebook is still useful. We

utilize these information by warm-start k-means with this codebook.

2. The norms of codewords reduce stage by stage. Though the naive "greedy" encoding fails to produce optimal encoding, a less "greedy" encoding is more likely to obtain the optimal encoding. We propose a multi-path encoding algorithm for learn codebooks.

The codebooks learned by IRVQ are mutually independent and each codebook has high information entropy. And a significantly lower quantization error observed compared to RVQ and other state-of-the-art methods. We have validated our method on two commonly used datasets for evaluating ANN search performance: SIFT-1M and GIST-1M (Jegou, Douze, and Schmid 2011). The empirical results show that our IRVQ improves RVQ significantly. Our IRVQ also outperforms other state-of-the-art quantization methods such as PQ, OPQ, and AQ.

## Residual Vector Quantization

Residual vector quantization (RVQ) (Juang and Gray Jr 1982) is a common technique to approximate original data with several low complexity quantizers, instead of a prohibitive high complexity quantizer. RVQ reduces the quantization error by learning quantizers on the residues. RVQ is introduced to perform ANN-search in (Chen, Guan, and Wang 2010). The gain of adding an additional stage relies on the commonality among residual vectors from different cluster centers. Thus on high-dimensional data this approach performs badly.

## Information Entropy

It has been observed that the residual vectors become very random with increasing stages, limiting the effectiveness of RVQ to a small number of stages. To begin with, we first examine the encoded dataset by RVQ from the point of view of information entropy.

For hashing based approximate nearest neighbor search methods, e.g. Spectral Hashing (Weiss, Torralba, and Fergus 2009), we seek a code that each bit has a 50% chance of being one or zero, and different bits are mutually independent. Similarly, we would like to obtain maximum information entropy  $S(\mathbf{C}_m)$ , defined below, for each codebook and no mutual information between different codebooks.

$$S(\mathbf{C}_m) = \sum_{k=1}^K p_k^m (\log_2 p_k^m) = \log_2 K$$

$$\sum_{k_i, k_j \in 1 \dots K} p_{ij}(k_i, k_j) \log_2 \frac{p_{ij}(k_i, k_j)}{p_{k_i}^i p_{k_j}^j} = 0 \quad (1)$$

for  $i, j \in 1 \dots M$

where  $p_k^m$  denotes the probability that in the dictionary  $\mathbf{C}_m$ , its  $k$ -th element is chosen; and  $p_{ij}(k_i, k_j)$  denotes the probability that  $k_i$ -th element from  $\mathbf{C}_i$  and  $k_j$ -th element from  $\mathbf{C}_j$  are chosen by a vector  $\mathbf{x}$  simultaneously. In Fig. 2, we drew the information entropy at each

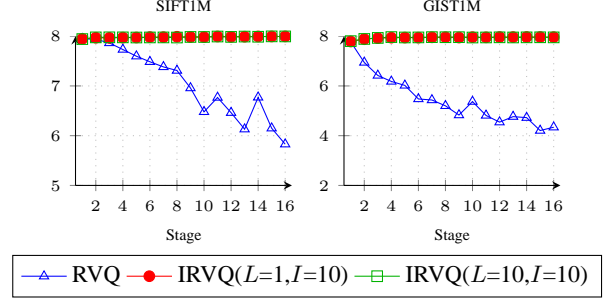


Figure 2: Entropy of each stage on SIFT1M and GIST1M datasets. Learned with  $k = 256$ . For the codebook learned on stage  $m$ , information entropy  $E = \sum_{n=1}^m p_n (\log_2 p_n)$ , where  $p_n = P(i_m(\mathbf{x}) = n)$ .

stage of the RVQ to reveal a phenomenon: For SIFT-1M dataset (Jegou, Douze, and Schmid 2011), the entropy drops to 5.8bits at the 16th stage, and 4.34bits for GIST-1M. That means though we wish to learn  $K$  codewords for each codebook, only a few of them are effective. Nevertheless, quantization on residual vectors leads to a much lower mutual information compared to PQ because residues are largely independent as shown in Fig.1.

## Encoding

RVQ encodes according to the distance between current residual vector and each codewords. We shall call it Sequential Encoding, which is actually a greedy algorithm. For codebooks learned with RVQ, The codewords chosen on a stage will affect the choice on the latter stages. Consider encoding a vector  $\mathbf{x}$  with  $M$  codebooks learned, the quantization error is:

$$E = \sum_{m=1}^{M'} \|\mathbf{x} - \mathbf{c}_m(i_m(\mathbf{x}))\|^2 - (m-1)\|\mathbf{x}\|^2$$

$$+ \sum_{a=1}^{M'} \sum_{b=1, a \neq b}^{M'} \mathbf{c}_a(i_a(\mathbf{x}))^T \mathbf{c}_b(i_b(\mathbf{x})) \quad (2)$$

Denote  $\epsilon$  as the third term. Sequential Encoding cannot predetermine the value of  $\epsilon$ . For PQ-based schemes such as Product Quantization (Jegou, Douze, and Schmid 2011), Optimized Product Quantization (Ge et al. 2013), such minimization scheme guarantees finding the best encoding, since codewords from different codebooks are mutually orthogonal. However, for RVQ, Sequential Encoding is not optimal. The encoding with codebooks learned by RVQ is actually a fully connected discrete pairwise MRF energy optimization, which is actually NP-hard.

In addition, the quantization errors are propagated to the next stage in the codebook learning process. RVQ actually uses the information of the encoded vectors on its learning phase: the residual vectors are determined by both codebooks and the encoding mechanism. As RVQ for ANN search requires many stages of codebook learning, the ac-

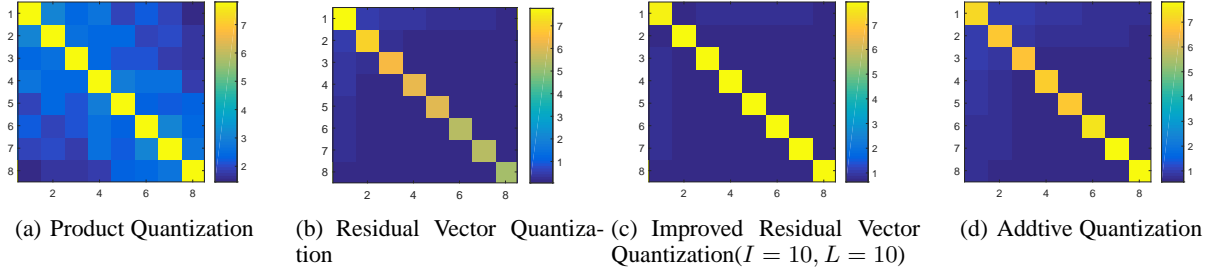


Figure 1: Mutual information matrix for different quantization methods. Experiment conducted on a GIST-1M dataset. We learned  $M = 8$  codebooks,  $K = 256$  codewords per codebook. The perfect encoding should have no mutual information between different codebooks and an information entropy of  $\log K = 8$ -bits for each codebook. Our proposed IRVQ achieves near optimal encoding.

cumulated quantization loss could become very high with Sequential Encoding.

### Improved Residual Vector Quantization

As we have presented the two major drawbacks of RVQ. In this section, we propose our Improved Revised Residual Vector Quantization (IRVQ) to overcome these issues. IRVQ improves RVQ in three aspects:

**Improved Codebook Learning** Instead of performing cold-started k-means as in RVQ, we warm-start k-means with initial codebook learned on a relatively smaller subspace.

**Multi-path Vector Encoding** In addition to assigning a vector to a single codeword, multiple codewords may be kept as candidates.

**Joint Optimization** On each stage, we first use Improved Codebook Learning to learn a codebook, then use Multi-path Vector Encoding to encode the dataset vectors, so the residual vectors could be even smaller.

See Algorithm 1 for the full pseudo code of IRVQ. The information entropy is maximized with our Improved Codebook Learning as shown in Figure 2, and with Multi-path Vector Encoding better approximations of original vectors are found as shown in Figure 3. We jointly use Improved Codebook Learning and Multi-path Vector Encoding to fully optimize the a stage, and the quantization error is even lowered. We shall explain our algorithm in more details in the following text.

### Improved Codebook Learning (ICL)

To obtain better clustering performance on high dimensional data, one of the popular approaches is to cluster on lower-dimensional subspace (Agrawal et al. 1998), this is also what PQ/OPQ do to obtain high information entropy for each dictionary. Many previously proposed methods for high dimensional data clustering, e.g. (Bouveyron, Girard, and Schmid 2007), (Jing, Ng, and Huang 2007), seek clustering in an optimal subspace instead of the whole feature space. In lower-dimensional subspaces the projected datasets become

---

### Algorithm 1 Improved Residual Vector Quantization

---

**Input:** Training samples  $\mathbf{X}$  of  $d$ -dimension, number of stages  $M$ , number of centroids to learn on each stage  $K$ , best  $L$  approximations on each stage,  $I$  iterations.

**Output:** Codebooks:  $\mathbf{C}_m = \{\mathbf{c}_m(1 \cdots K), m = 1 \cdots M\}$ .

- 1: Initialize Residue:  $\mathbf{R} = \mathbf{X}$
  - 2: Denote the best  $L$  approximations for  $\mathbf{x}$  on stage  $m$ :  $\{\mathbf{x}_m^l, l = 1 \cdots L\}$
  - 3: **for**  $m = 1 \cdots M$  **do**
  - 4:   Perform PCA on  $\mathbf{R}$  and extract principal components  $\mathbf{A} = (\mathbf{r}_1, \mathbf{r}_2, \cdots, \mathbf{r}_d)$ .
  - 5:    $\mathbf{C}'_m \leftarrow$  k-means on  $(\mathbf{r}_1, \mathbf{r}_2, \cdots, \mathbf{r}_{\lceil d^{1/I} \rceil})^T \mathbf{R}$
  - 6:   **for**  $p = 2 \cdots I$  **do**
  - 7:      $\mathbf{Y} \leftarrow (\mathbf{r}_1, \mathbf{r}_2, \cdots, \mathbf{r}_{\lceil d^{p/I} \rceil})^T \mathbf{R}$
  - 8:      $\mathbf{C}'_m \leftarrow$  k-means on  $\mathbf{Y}$ :  
       Initialize k-means algorithm with  $\mathbf{C}'_m$ , missing dimensions are padded with zeros.
  - 9:   **end for**
  - 10:    $\mathbf{C}_m = \mathbf{A} \mathbf{C}'_m$  is the codebook for stage  $m$ .
  - 11:   **for all**  $\mathbf{x} \in \mathbf{X}$  **do**
  - 12:     find the best  $L$  approximations of  $\mathbf{x} \approx \mathbf{x}_m^l = \mathbf{x}_{m-1}^{l'} + \mathbf{c}_m(k), l' = 1 \cdots L, k = 1 \cdots K$ .
  - 13:   **end for**
  - 14:    $\mathbf{r} = \mathbf{x} - \mathbf{x}_m^1$
  - 15: **end for**
- 

denser and then a balanced clustering could be easily obtained. However, in the case of fitting the residual vectors, it's not reasonable to clustering on just a few dimensions as the residue vectors lies in the whole feature space randomly. We thus seek a hybrid way to perform clustering on the residual vectors.

We propose Improved Codebook Learning (ICL) to this end. It is done by iteratively adding dimensions for clustering. First designate a dimensions adding sequenc:  $d_1 < d_2 < \cdots < d_I = d$ , then, then:

**Initialization** We first perform the principle component analysis (PCA) on the residual vectors  $\mathbf{R}$  and extract the principal dimensions.

**Iterative Warm-start** On the  $i$ -th iteration, perform k-means on top  $d_i$  dimensions, initialized with previous learn codewords<sup>1</sup>. Note that the codewords have missing dimensions, we simply pad zeros to them.

**Restoration** Transform the codebook in PCA dimensions back to the original data-space dimensions. Then the resulting codebook can be used in the same way as the codebooks learned with usual k-means algorithm.

There are several motivations for our improved codebook learning method:

1. As depicted in (Ding and He 2004), PCA dimension reduction finds the best low-dimensional L2 approximation of the data.
2. Finding a better initial points for k-means almost always leads to a quicker convergence and a better clustering (Bradley and Fayyad 1998). As padding extra dimensions with zeros doesn't affect the encoding, it's reasonable to infer that the iterative initialization could also lead to better clustering.
3. PCA dimensions are essentially rotation of original data with no loss of information or metric change. Padding zeros doesn't change the clustering results.

We warm-start k-means on the most significant PCA dimensions so clustering on the higher dimensions could started from better initial positions. In our experiment, a very low  $I$  (in our experiments,  $I=10$  at most), could already achieve satisfying results. By performing ICL, As shown in Figure 2, the entropy of each codebook is maximized.

### Multi-path Vector Encoding (MVE)

Encoding for Product Quantization is quite simple since the original feature space has been divided into mutually orthogonal subspaces. However, for Additive Quantization (Barnes and Frost 1993), Composite Quantization (Ting Zhang 2014), and Residual Vector Quantization, the encoding is an MRF problem as mentioned in Section . Additive Quantization (Babenko and Lempitsky 2014) proposed a Beam Search algorithm in a matching pursuit fashion, however, runs slowly in practice (Babenko and Lempitsky 2015).

Suppose the best approximation (correct encoding) of an input vector is  $\mathbf{x} \approx \mathbf{c}_1(i_1) + \mathbf{c}_2(i_2) + \dots + \mathbf{c}_M(i_M)$ . Further assume we have known the first  $m-1$  correct encodings  $i_1, i_2, \dots, i_{m-1}$ , can we effectively compute  $i_m$ ? Denote the known part as  $\hat{\mathbf{x}} = \mathbf{c}_1(i_1) + \dots + \mathbf{c}_{m-1}(i_{m-1})$  and the unknown part as  $\tilde{\mathbf{x}} = \mathbf{c}_{m+1}(i_{m+1}) + \dots + \mathbf{c}_M(i_M)$ , we seek the correct encoding on the  $m$ -th dictionary  $i_m$ . Notice that:

$$\begin{aligned} \|\mathbf{x} - \hat{\mathbf{x}} - \mathbf{c}_m(i_m) - \mathbf{x}'\|^2 &= \|\mathbf{x} - \hat{\mathbf{x}}\|^2 + \|\mathbf{x} - \tilde{\mathbf{x}}\|^2 + 2\hat{\mathbf{x}}^T \tilde{\mathbf{x}} \\ &\quad + \|\mathbf{x} - \mathbf{c}_m(i_m)\|^2 + 2\hat{\mathbf{x}}^T \mathbf{c}_m(i_m) + 2\mathbf{c}_m(i_m)^T \tilde{\mathbf{x}} \\ &\quad - 2\|\mathbf{x}\|^2 \end{aligned} \quad (3)$$

<sup>1</sup>On the first iteration we directly learn a codebook

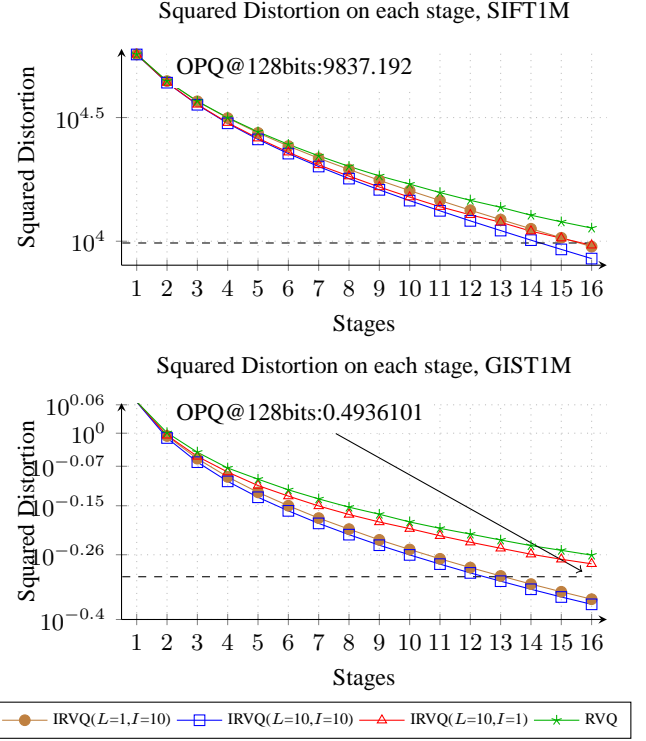


Figure 3: Distortion on SIFT1M and GIST1M, with  $M = 16$ ,  $k = 256$ , also compared with Optimized Product Quantization. Note that  $L=10$   $I=1$ : use MVE only;  $L=1$ ,  $I=10$ : use ICL only;  $L=10$ ,  $I=10$ : Jointly use MVE and ICL

The first three terms can be seen as constants when we seek the correct  $i_m$ , and the last term can be omitted. The fourth and fifth term can be effectively computed. However the sixth term cannot be computed because we don't know  $\tilde{\mathbf{x}}$ . If we omit this term extra error will be introduced. To lessen this error, we hope  $\|\tilde{\mathbf{x}}\|$  is very small so that the variance of the last term won't have a serious impact on the final outcome. The norms of codewords from codebooks learned with IRVQ naturally shrinks, so  $\|\tilde{\mathbf{x}}\|$  is decreased stage by stage.

Thus we maintain a list of best  $L$  approximations of  $\mathbf{x}$  with the first  $(m-1)$  codebooks:  $\{\mathbf{x}_{m-1}^1, \mathbf{x}_{m-1}^2, \dots, \mathbf{x}_{m-1}^L\}$ . Then we encode with the next codebook  $\mathbf{C}_m = \{\mathbf{c}_m(1), \mathbf{c}_m(2), \dots, \mathbf{c}_m(K)\}$ . We find  $L$  combinations from  $\{\mathbf{x}_{m-1}^l + \mathbf{c}_m(k)\}$ ,  $l \in 1 \dots L, k \in 1 \dots K$  minimizing the following objective function:

$$\begin{aligned} \|\mathbf{x} - \mathbf{x}_{m-1}^l - \mathbf{c}_m(k)\|^2 &= \|\mathbf{x} - \mathbf{x}_{m-1}^l\|^2 + \|\mathbf{x} - \mathbf{c}_m(k)\|^2 \\ &\quad - \|\mathbf{x}\|^2 + 2\mathbf{c}_m(k)^T \mathbf{x}_{m-1}^{l-m-1} \end{aligned} \quad (4)$$

The first term has been computed at the previous encoding step, and the third term  $\|\mathbf{x}\|^2$  is constant for any  $(\mathbf{x}_{m-1}^l + \mathbf{c}_m(k))$ , is thus negligible. And the last term involves  $m$  table lookups and addition, with the inner-product of all codewords precomputed before MVE. Thus, only the

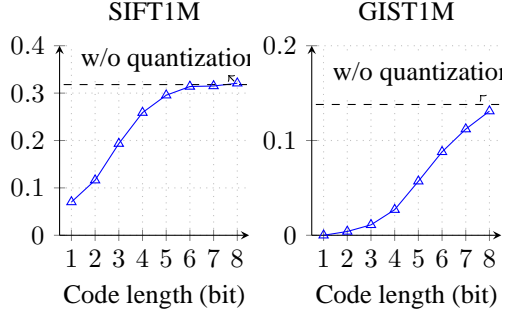


Figure 4: How quantization on  $\epsilon$  affects ANN search performance (Measured by Recall@1). Experimented on SIFT-1M and GIST-1M datasets, with  $M = 8, k = 256, L = 30, I = 10$

term  $\|\mathbf{x} - \mathbf{c}_m(k)\|^2$  is required to be computed. The time complexity is  $O(dK + mKL + KL \log L)$  for encoding with one single dictionary on the  $m$ -th stage.

To sum up, MVE iteratively uses the top  $L$  candidates as seeds to find the best encoding for  $\mathbf{x}$ . The resulting method is quite similar to the multi-path search for residual tree (Kossentini, Smith, and Barnes 1992), which is used on signal reconstruction. Note that MVE degrades to Sequential Encoding when  $L = 1$ .

### Learning codebooks jointly with Improved Codebook Learning and Multi-path Encoding

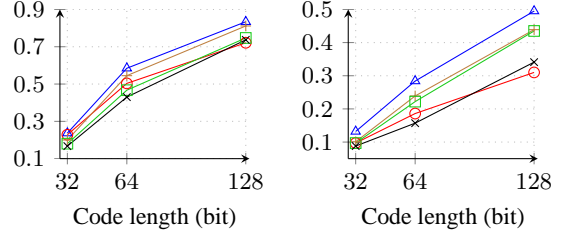
Remind that RVQ is a multiple stage learning procedure, at each stage the quantization error is left to the next iteration. This makes reducing quantization error for every single stage necessary. Now we have already presented the Improved Codebook Learning for learning balanced code-words, and Multi-path Vector Encoding for better encoding the original vector, we next jointly use them to optimize a stage in IRVQ. At each stage, our IRVQ does the following:

1. Use the Improved Codebook Learning to learn a codebook;
2. Use the Multi-path Vector Encoding to find a best approximation of the original vector with all the codebooks learned.

Note that for each stage, the  $L$  best approximations generated by MVE can be stored for future use, so the MVE on next stage only requires to encode one more codebook.

### Further Lower the Memory Consumption

One of the advantage of quantization methods for ANN search is that vectors are compressed into a few bits so an in-memory search is feasible. Take PQ as an example, for an encoding with  $M = 8, K = 256$ , an original vector could be compressed into  $M \log_2 K = 64$ -bit code, thus a dataset containing 1 billion vectors can be fully loaded into a 8G RAM. For RVQ, AQ and our IRVQ, an extra storage overhead is required. Consider the expansion of distance be-



(a) Recall@4 on SIFT1M, (b) Recall@4 on GIST1M, different code length



Figure 6: Effect of different code length for different ANN search methods. Experimented on datasets SIFT1M and GIST1M. The recall of the true neighbor in the top 4 ranked quantized element (Recall@4) is used to measure the ANN search quality.

tween the approximated vector  $\hat{\mathbf{x}}$  and the query vector  $\mathbf{q}$ :

$$\begin{aligned} \|\mathbf{q} - \hat{\mathbf{x}}\|^2 &= \sum_{m=1}^M \|\mathbf{q} - \mathbf{c}_m(i_m(\mathbf{x}))\|^2 - (m-1)\|\mathbf{q}\|^2 + \epsilon \\ \epsilon &= \sum_{a=1}^M \sum_{b=1, b \neq a}^M \mathbf{c}_a(i_a(\mathbf{x}))^T \mathbf{c}_b(i_b(\mathbf{x})) \end{aligned} \quad (5)$$

Note that  $\epsilon$  is irrelevant to query  $\mathbf{q}$ , so we compute and store  $\epsilon$  along with each quantized vector when the whole database is quantized, and this  $\epsilon$  causes extra memory consumption. An IEEE-754 floating point number takes 32-bit of memory. However, it's possible to quantize  $\epsilon$  into a few bits. Figure 4 shows that using only 8-bits of extra storage could already preserve the searching quality.

For real world ANN search applications, the major concern is the response speed instead of the memory consumption. Since an additional table look-up (indirect addressing) takes much more time compared to memory copy, if memory size is not an issue we recommend not to quantize  $\epsilon$  for fastest speed.

### Approximate Nearest Neighbor Search Performance

We performed the ANN search tests on the two datasets commonly used to validate the efficiency of ANN methods: SIFT-1M and GIST-1M from (Jegou, Douze, and Schmid 2011):

**SIFT1M** contains one million of 128-d SIFT (Lowe 2004) features. It's commonly used with local feature descriptor for various image related applications.

**GIST1M** contains one million of 960-d GIST (Oliva and Torralba 2001) global descriptors.

For each dataset, we randomly pick 100,000 vectors as the training set. We then encode the rest of the database vectors, and perform 1000 queries to check ANN search quality.



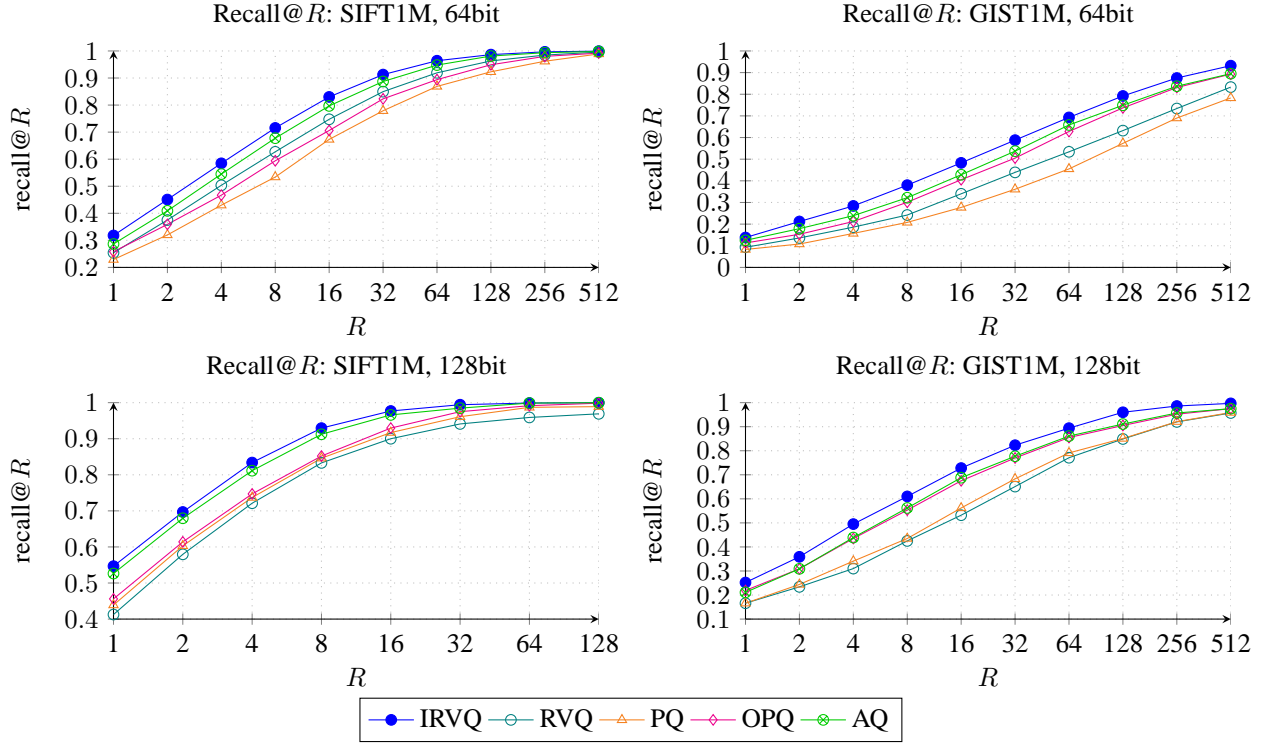


Figure 5: The performance for different algorithms on SIFT-1M and GIST-1M, with 128-bit encoding( $M=16$ )

## Evaluated Methods

We compared our IRVQ to the following state-of-the-art quantization methods:

**PQ** : Product quantization proposed in (Jegou, Douze, and Schmid 2011). Following (Jegou, Douze, and Schmid 2011), we used the structured ordering for GIST-1M and the natural ordering for SIFT-1M.

**OPQ** : Optimized Product Quantization proposed in (Ge et al. 2013). We adopted the non-parametric version of OPQ. Cartesian k-means, the algorithm proposed in (Norouzi and Fleet 2013) shares a similar idea and has the same performance with OPQ.

**AQ** : Additive Quantization (Babenko and Lempitsky 2014). We adopted the parameters suggest in the paper.

**RVQ** : Residual Vector Quantization proposed in (Chen, Guan, and Wang 2010).

For our IRVQ we set the parameters as  $I = 10$ ,  $L = 30$ . For all the methods, we choose  $K = 256$  as the size of each codebook. We perform linear scan search with asymmetric distances computation (ADC) proposed in (Jegou, Douze, and Schmid 2011), which directly compares the input query and the quantized dataset. The search quality is measured using  $\text{recall}@R$ , which means that for each query, we retrieved  $R$  nearest items and check whether they

contain the true nearest neighbor. Such criterion is commonly used to evaluate the ANN methods.

## Results

Figure 5 shows the recall curve of various state-of-the-art methods. Our IRVQ improves RVQ significantly, for example, on 64bit encoding, IRVQ obtained **58.31%** recall@4 for SIFT1M, while plain RVQ is only 50.35%, the relative improvement is **15.8%**. The improvement is even significant on higher dimensional data GIST1M, where IRVQ gained **28.4%** and RVQ gained 18.6% accuracy, relatively **52.7%** improvement on recall@4. IRVQ also outperforms other state-of-the-art, for example, IRVQ outperforms AQ by **17.7%** on the recall@1 for 64bit SIFT1M encoding.

We can also see by the results, RVQ doesn't perform well with added stages. On lower bits and low dimensions RVQ has advantages over PQ/OPQ, however, on higher bits or higher dimensions plain RVQ performs badly. Fig.6 illustrates the effect of code length for different methods. Our IRVQ fixed the problem and deliver consistently high performance gain with added stages.

## Conclusion

In this paper, we proposed the Improved Residue Vector Quantization (IRVQ) for large-scale high-dimensional approximate nearest neighbor search. We proposed Improved Codebook Learning (RCL) and the Multi-path Vector Encoding (MVE) to deliver consistent performance gain with

adding stages. Experiment against several state-of-the-art quantization methods on two well known dataset demonstrate the effectiveness of IRVQ.

## References

- [Agrawal et al. 1998] Agrawal, R.; Gehrke, J.; Gunopulos, D.; and Raghavan, P. 1998. *Automatic subspace clustering of high dimensional data for data mining applications*, volume 27. ACM.
- [Babenko and Lempitsky 2014] Babenko, A., and Lempitsky, V. 2014. Additive quantization for extreme vector compression. In *Computer Vision and Pattern Recognition (CVPR), 2014 IEEE Conference on*, 931–938. IEEE.
- [Babenko and Lempitsky 2015] Babenko, A., and Lempitsky, V. 2015. Tree quantization for large-scale similarity search and classification. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 4240–4248.
- [Barnes and Frost 1993] Barnes, C. F., and Frost, R. L. 1993. Vector quantizers with direct sum codebooks. *Information Theory, IEEE Transactions on* 39(2):565–580.
- [Bouveyron, Girard, and Schmid 2007] Bouveyron, C.; Girard, S.; and Schmid, C. 2007. High-dimensional data clustering. *Computational Statistics & Data Analysis* 52(1):502–519.
- [Bradley and Fayyad 1998] Bradley, P. S., and Fayyad, U. M. 1998. Refining initial points for k-means clustering. In *ICML*, volume 98, 91–99. Citeseer.
- [Chen, Guan, and Wang 2010] Chen, Y.; Guan, T.; and Wang, C. 2010. Approximate nearest neighbor search by residual vector quantization. *Sensors* 10(12):11259–11273.
- [Ding and He 2004] Ding, C., and He, X. 2004. K-means clustering via principal component analysis. In *Proceedings of the twenty-first international conference on Machine learning*, 29. ACM.
- [Ge et al. 2013] Ge, T.; He, K.; Ke, Q.; and Sun, J. 2013. Optimized product quantization for approximate nearest neighbor search. In *Computer Vision and Pattern Recognition (CVPR), 2013 IEEE Conference on*, 2946–2953. IEEE.
- [Gersho and Gray 1992] Gersho, A., and Gray, R. M. 1992. *Vector quantization and signal compression*. Springer Science & Business Media.
- [Indyk and Motwani 1998] Indyk, P., and Motwani, R. 1998. Approximate nearest neighbors: towards removing the curse of dimensionality. In *Proceedings of the thirtieth annual ACM symposium on Theory of computing*, 604–613. ACM.
- [Jegou, Douze, and Schmid 2011] Jegou, H.; Douze, M.; and Schmid, C. 2011. Product quantization for nearest neighbor search. *Pattern Analysis and Machine Intelligence, IEEE Transactions on* 33(1):117–128.
- [Jing, Ng, and Huang 2007] Jing, L.; Ng, M. K.; and Huang, J. Z. 2007. An entropy weighting k-means algorithm for subspace clustering of high-dimensional sparse data. *Knowledge and Data Engineering, IEEE Transactions on* 19(8):1026–1041.
- [Juang and Gray Jr 1982] Juang, B.-H., and Gray Jr, A. 1982. Multiple stage vector quantization for speech coding. In *Acoustics, Speech, and Signal Processing, IEEE International Conference on ICASSP'82.*, volume 7, 597–600. IEEE.
- [Kossentini, Smith, and Barnes 1992] Kossentini, F.; Smith, M. J.; and Barnes, C. 1992. Large block rvq with multipath searching. In *Circuits and Systems, 1992. ISCAS'92. Proceedings., 1992 IEEE International Symposium on*, volume 5, 2276–2279. IEEE.
- [Kossentini, Smith, and Barnes 1995] Kossentini, F.; Smith, M. J.; and Barnes, C. F. 1995. Image coding using entropy-constrained residual vector quantization. *Image Processing, IEEE Transactions on* 4(10):1349–1357.
- [Lowe 1999] Lowe, D. G. 1999. Object recognition from local scale-invariant features. In *Computer vision, 1999. The proceedings of the seventh IEEE international conference on*, volume 2, 1150–1157. Ieee.
- [Lowe 2004] Lowe, D. G. 2004. Distinctive image features from scale-invariant keypoints. *International journal of computer vision* 60(2):91–110.
- [Norouzi and Fleet 2013] Norouzi, M., and Fleet, D. J. 2013. Cartesian k-means. In *Computer Vision and Pattern Recognition (CVPR), 2013 IEEE Conference on*, 3017–3024. IEEE.
- [Oliva and Torralba 2001] Oliva, A., and Torralba, A. 2001. Modeling the shape of the scene: A holistic representation of the spatial envelope. *International journal of computer vision* 42(3):145–175.
- [Rui, Huang, and Chang 1999] Rui, Y.; Huang, T. S.; and Chang, S.-F. 1999. Image retrieval: Current techniques, promising directions, and open issues. *Journal of visual communication and image representation* 10(1):39–62.
- [Ting Zhang 2014] Ting Zhang, Chao Du, J. W. 2014. Composite quantization for approximate nearest neighbor search. *Journal of Machine Learning Research: Workshop and Conference Proceedings* 32(1):838–846.
- [Weiss, Torralba, and Fergus 2009] Weiss, Y.; Torralba, A.; and Fergus, R. 2009. Spectral hashing. In *Advances in neural information processing systems*, 1753–1760.